DIMSG Array

A main source of the DIMSG array is output from the Loader: EN^DDIOL. Writes that are currently embedded in the database must be changed to calls to EN^DDIOL if the DBS is to be used. When running applications in scrolling mode, the Loader simply Writes the text to the screen. However, if the node containing the EN^DDIOL call is executed from within one of the DBS calls, the DBS returns text in an array, usually subscripted by DIMSG. (For more detailed information about EN^DDIOL, see its description in the Classic FileMan API section of this manual.)

When the user is not in scrolling mode, the Loader will most frequently place the text into the DIMSG array with the local variable DIMSG set equal to the total number of lines in the array. There are certain situations, however, where the output is put into another array. As mentioned above, when the DBS HELP^DIE call is used to get help, the output of an EN^DDIOL call embedded in Xecutable Help is placed into the DIHELP array.

Like DIHELP, the DIMSG array is simply a list of lines of text.

Suppose an INPUT transform currently contains:

```
N Y S Y=$L(X) K:Y>30!(Y<3) X I '$D(X) W !,"Your input was "_Y_
" characters long.",!,"This is the wrong length."
```

It can be changed to:

```
N Y S Y=$L(X) K:Y>30!(Y<3) X I '$D(X) S Y(1)="Your input was "_Y_
" characters long.",Y(2)="This is the wrong length." D EN^DDIOL(.Y)
```

This change would have no effect if the user were in scrolling mode; the same message is written to the screen. However, if the second INPUT transform were executed from a silent call, nothing is Written and the "DIMSG" array returned to the client application might look like this:

```
^TMP("DIMSG",$J,1)="Your input was 2 characters long."
^TMP("DIMSG",$J,2)="This is the wrong length."
```

DIERR Array

When an error condition is encountered during a DBS call, an error message and other information is placed in the DIERR array. In addition, the DIERR variable is returned with the following two pieces of information: the number of errors generated during the call in the first piece and the total number of lines of the error

messages in the second. Thus, a $D check on the variable DIERR after the completion of the call allows the client application to determine if an error occurred. Both syntactical (e.g., the root of an array is not in the proper format for subscript indirection) and substantive substantive (e.g., a specified field does not exist in the specified file) errors are returned.

The information contained in the DIERR array is designed to give the client application specific information about the kind of error that occurred to allow for intelligent error handling and to provide readable error messages. Here is an example of error reporting following a Filer call:

```
>W $G(DIERR)
2^2
>D ^%G

Global ^TMP("DIERR",$J
        TMP("DIERR",$J
^TMP("DIERR",731990208,1) = 305
^TMP("DIERR",731990208,1,"PARAM",0) = 1
^TMP("DIERR",731990208,1,"PARAM",1) = ^TMP("MYWPDATA",$J)
^TMP("DIERR",731990208,1,"TEXT",1) = The array with a root of
   '^TMP("MYWPDATA",$J)' has no data associated with it.
^TMP("DIERR",731990208,2) = 501
^TMP("DIERR",731990208,2,"PARAM",0) = 3
^TMP("DIERR",731990208,2,"PARAM",1) = 89
^TMP("DIERR",731990208,2,"PARAM","FIELD") = 89
^TMP("DIERR",731990208,2,"PARAM","FILE") = 16200
^TMP("DIERR",731990208,2,"TEXT",1) = File #16200 does not contain
   a field 89.
^TMP("DIERR",731990208,"E",305,1) =
^TMP("DIERR",731990208,"E",501,2) =
```

The DIERR variable acts like a flag. In the example above, it reports that two errors occurred and that they have a total of two lines of text.

The ^TMP("DIERR",$J) global contains information about the error(s).

```
^TMP("DIERR",$J,sequence#) = error number
```

In this case, two errors were returned: errors #305 and #501. Each error number corresponds to an entry in the DIALOG file. The actual text of each error is stored in nodes descendent from "TEXT":

```
^TMP("DIERR",$J,sequence#,"TEXT",line#) = line of text
```

The ^TMP("DIERR",$J,sequence#,"PARAM") subtree contains specific parameters that may be returned with each error:

```
^TMP("DIERR",$J,sequence#,"PARAM",0) = number of parameters returned with
                                       the error
```

```
^TMP("DIERR",$J,sequence#,"PARAM","param_name") = parameter value
```

The VA FileMan error messages and their associated parameters are documented in Appendix A-VA FileMan Error Codes in this manual. For example, Appendix A indicates that three parameters are returned with error #501: '1', the field name or number; 'FILE', the File number; and 'FIELD', the Field number. So, in the example above, for error #501, the "PARAM" nodes indicate that the error corresponds to File #16200, Field #89.

Finally, the "E" cross-reference in the ^TMP("DIERR",$J) global allows you to determine quickly whether a particular error occurred. For example, if you wanted to do some special error processing if a DBS call generated error #305, you could check $D(^TMP("DIERR",$J,"E",305)).

The DIERR array is more complicated than the other arrays discussed, thereby making more information available to the client application for error handling.

Obtaining Formatted Text From The Arrays

If you want the text from any of the three arrays, the following call extracts it from the structures described above and either writes it to the screen or puts it into a local array for further use:

```
D MSG^DIALOG(FLAGS,.OUTPUT_ARRAY,TEXT_WIDTH,LEFT_MARGIN,
  INPUT_ROOT)
```

The flags for this call control whether the text is Written to the current device or moved into the output_array specified in the second parameter. The flags also direct whether the source arrays are saved or deleted and which kinds of dialog (errors, help, or other messages) are processed. Some formatting of text is also supported. See the description of MSG^DIALOG in this DBS section for details of its use.

Cleaning Up the Output Arrays

When you make a DBS call and use the default arrays in the ^TMP global for output of help, user, and error messages, the DBS call kills off these arrays and their related variables at the start of the call. Therefore, you know that any data that exists after the call was generated by that call.

If you don't use the default arrays for output, however, and instead specify your own arrays for this information to be returned in, your arrays are not automatically

killed at the start of a DBS call. So if there is any chance that these arrays might already exist, you should kill them yourself before making the DBS call.

After making a DBS call, if you used the default arrays in ^TMP for output of help, user, and error messages, you should delete these arrays before your application Quits. To do this, use the following call:

```
D CLEAN^DILF
```

See the description of CLEAN^DILF later in this DBS section for details of its use.

If you are using your own arrays for output, however, you need to clean up your arrays yourself. You should still call CLEAN^DILF to kill off the variables related to these arrays, however.

Example of Call to VA FileMan DBS

One of the DBS calls validates data. If the data is valid, the internal representation of that data is returned. If the data is invalid, an up-arrow (^) is returned along with various messages, optionally including the relevant help text. The validate call looks like this (see the Validator documentation for details):

```
VAL^DIE(FILE,IENS,FIELD,FLAGS,VALUE,.RESULT,FDA_ROOT,MSG_ROOT)
```

Your call might look like this:

```
D VAL^DIE(999000,"223,",4,"H","AB",.MYANSWER,"","MYMSGS(""WIN3"")")
```

If MYANSWER equaled "^" after the call, your MYMSGS("WIN3") array might look like :

```
MYMSGS("WIN3","DIERR")=1^1
MYMSGS("WIN3","DIERR",1)=701
MYMSGS("WIN3","DIERR",1,"PARAM",0)=4
MYMSGS("WIN3","DIERR",1,"PARAM",3)="AB"
MYMSGS("WIN3","DIERR",1,"PARAM","FIELD")=4
MYMSGS("WIN3","DIERR",1,"PARAM","FILE")=999000
MYMSGS("WIN3","DIERR",1,"PARAM","IENS")="223"
MYMSGS("WIN3","DIERR",1,"TEXT",1)="The value 'AB' for field ALPHA
  DATA in file TEST1 is not valid."
MYMSGS("WIN3","DIERR","E",701,1)=""
MYMSGS("WIN3","DIHELP")=1
MYMSGS("WIN3","DIHELP,1)="Answer must be 3-30 characters in length."
MYMSGS("WIN3","DIMSG")=1
MYMSGS("WIN3","DIMSG",1)="Your input was 2 characters long."
MYMSGS("WIN3","DIMSG",2)="This is the wrong length."
```

The DIERR portion of this array indicates that error number 701 is being reported. Documentation makes clear that this means that an input value was invalid. The PARAM nodes (also documented) give the client application the relevant file#, field#, IENS, and value. This information might be used by the application in its error handling. The TEXT node contains the error message; note that it is customized to include specifics of the current error. The DIHELP node contains single-question-mark help for the field. The DIMSG nodes contain a message generated by the INPUT transform via an EN^DDIOL call. (The sample INPUT transform discussed in the DIMSG section above produced this message.)

Now, the client application decides what (if anything) to show the user. In a GUI environment, you might decide to put the error message along with any text from the INPUT transform into a document gadget. A HELP button that could be used by the user to display the help information might be added to the box. FileMan's DBS has provided text; the client application is in complete control regarding the use of this text.